webir

Clase 5

webir - Temas

- Repaso: Construcción de índices
  - Indexación Basada en Ordenamiento por Bloques (BSBI)
  - Indexación de un Sólo Pasaje en Memoria (SPIMI)
- Construcción de índices – algoritmos distribuidos
- Compresión de índices: vocabulario o lista de postings

webir - Construcción de Índices - Proceso

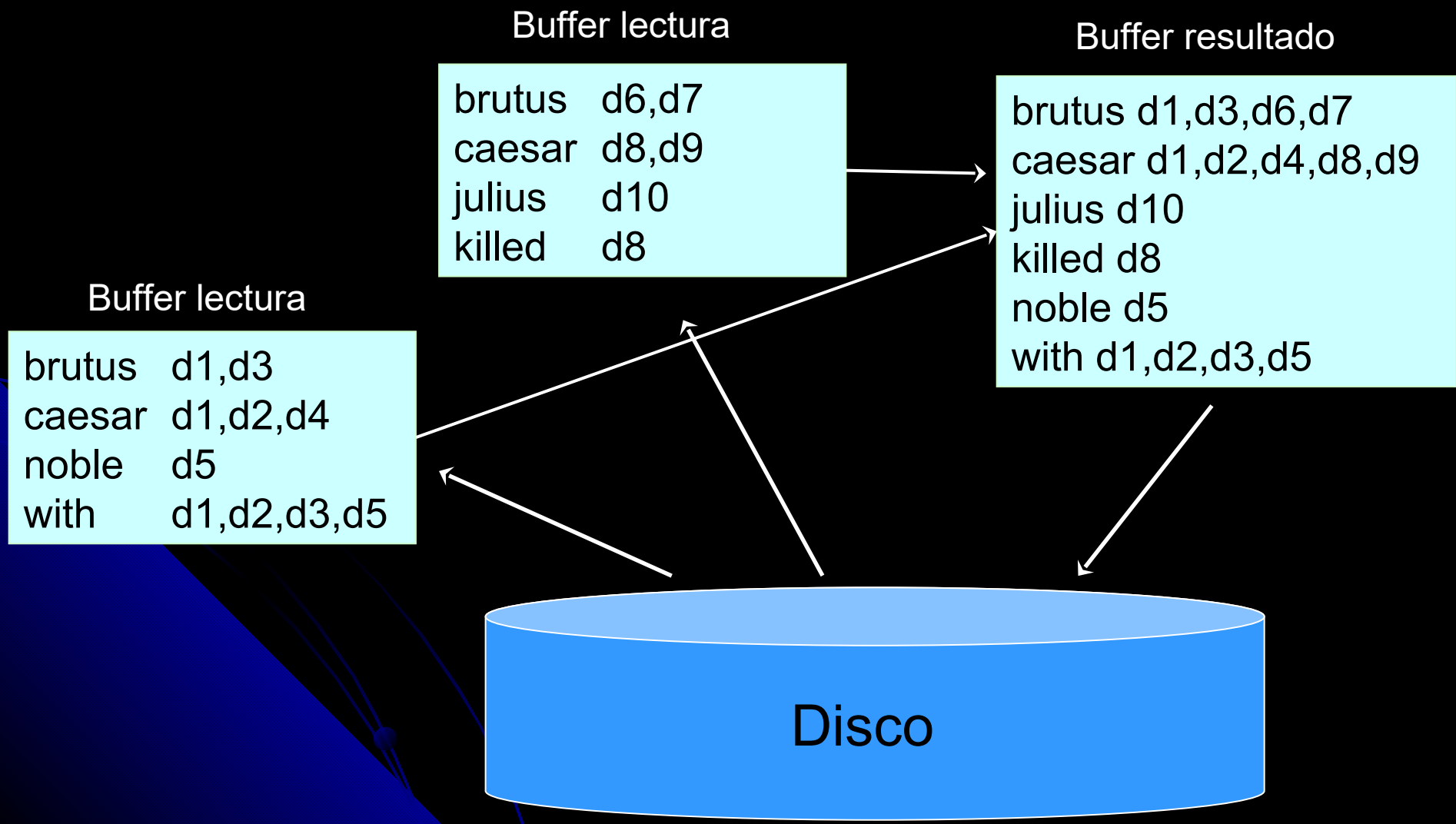
- Colección de documentos a indexar
- Separar en palabras (tokenize)
- Procesamiento lingüístico para normalizar las palabras
- Ordenar las palabras en orden alfabético
  - pares de términos (clave primaria) y docID (clave secundaria)
- Unificar ocurrencias repetidas de palabras
- Ordenar postings por docID (clave secundaria)

webir - Indexación Basada en Ordenamiento por Bloques (BSBI)

1. Dividir la colección en porciones de igual tamaño (de tal forma que quede holgura en memoria)
2. Generar índice invertido de cada porción
    1. Obtener conjunto de pares (termID, docID)
    2. Ordenar los pares (termID, docID)
    3. Unificar ocurrencias repetidas de los términos
3. Guardar resultados parciales en disco
4. Unificar los resultados parciales (merge)
    - Abrir todos los bloques en simultáneo
    - Tener un pequeño buffer de lectura para cada bloque y uno para la escritura del resultado
    - En cada iteración elegir el menor termID, usar PQ o similar

Bloques contiguos para minimizar tiempos de seek.

webir - Indexación Basada en Ordenamiento por Bloques (BSBI)



webir - Indexación de un Sólo Pasaje en Memoria (SPIMI)

- En lugar de termID se usan los términos
- Algoritmo
  1. Dividir la colección en porciones de igual tamaño
  2. Generar índice invertido de cada porción procesando cada término en orden
     1. Si es la primera ocurrencia, agregarlo al diccionario parcial (hash)
     2. Agregar directamente docID a la lista de postings del término, si es necesario agrandar el espacio para la lista (no se guardan las (term, docID como en BSBI))
  3. Guardar resultados parciales ordenados en disco – en orden lexicográfico
  4. Unificar los resultados parciales (merge) igual que BSBI

webir - Indexación de un Sólo Pasaje en Memoria (SPIMI)

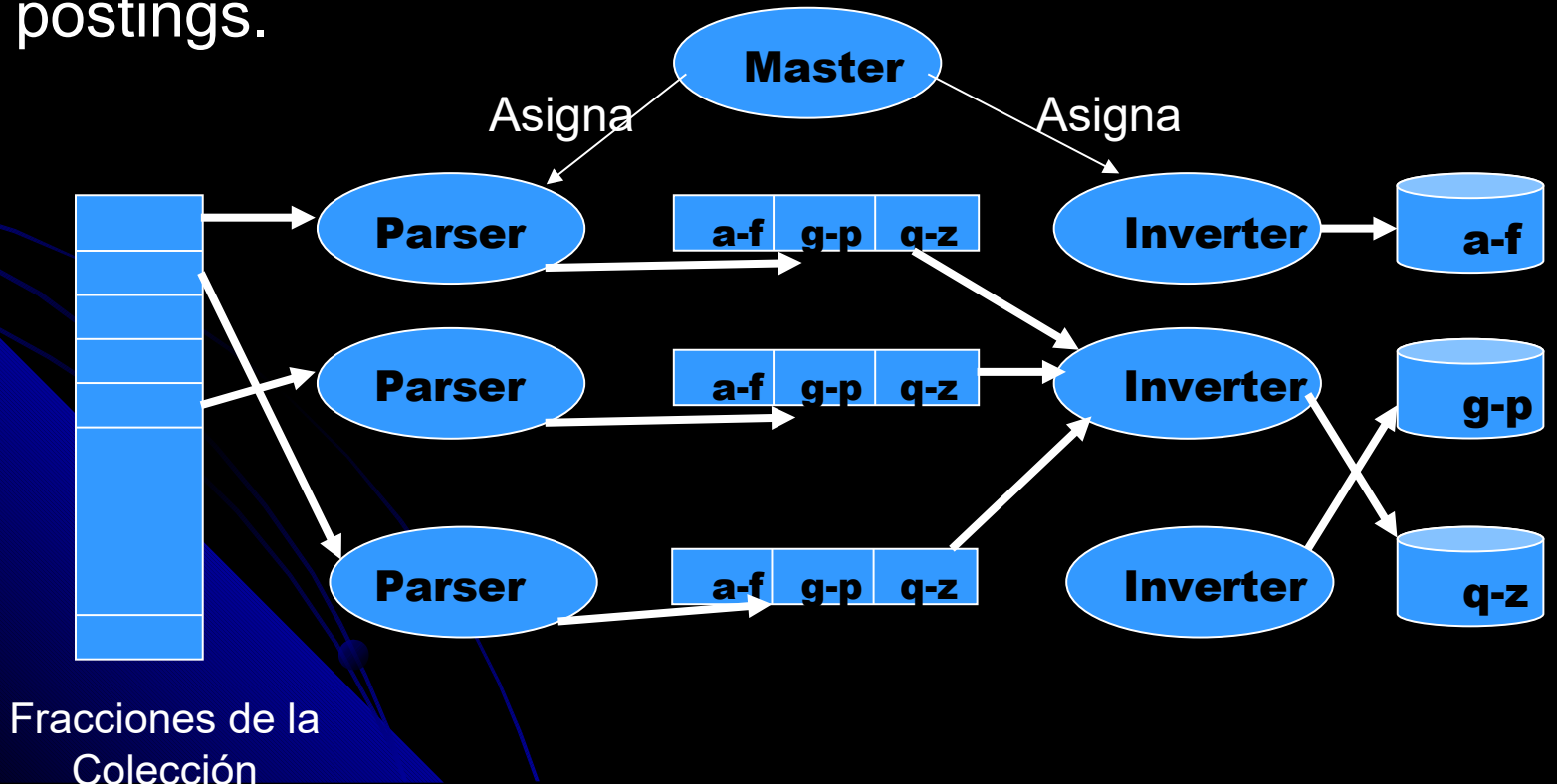
- BSBI - O(T log T)
  - Debido al ordenamiento de los pares (termID, docID)
  - En general dominado por
    - Tokenization y el procesamiento lingüístico
    - Unificación de resultados parciales
- SPIMI - SPIMI es O(T)
  - SPIMI agrega directamente el docID a la lista de postigs sin agrupar y luego ordenar
    - SPIMI no requiere ordenamiento – más rápido
  - No guarda pares (termID, docID) – menos memoria

webir - Indexación Distribuida – Términos

- Índices distribuidos, grandes colecciones
  - Resultados también distribuidos, fraccionado por
    - Términos
    - Documentos (más frecuente)
- Arquitectura distribuida MapReduce
  - Varias computadoras estándar - nodos
  - Google “MapReduce: Simplified Data Processing on Large Clusters ”
  - La tarea inicial (Map) es de parsing, al igual que en BSBI y SPIMI, se genera un diccionario parcial en cada nodo, dividido en segmentos por ejemplo a-f, g-p y q-z
  - Luego (Reduce) se genera una sola lista de postings para cada termID (Invert), cada segmento es asignado a un nodo

webir - Ejercicios

- En la etapa de “invertir” en la indexación distribuida usando MapReduce, se debe distribuir de forma equitativa las fracciones del índice invertido a construir de modo tal que contengan aproximadamente la misma cantidad de postings.



webir - Indexación Dinámica

- Agregar/quitar términos
- Actualizar listas de postings

- Reconstruir completamente el índice
  - Pocos/espóradicos cambios
  - Tiempo de recontrucción completa es aceptable para incorporar nuevos documentos al índice
  - Recursos suficientes para reconstruir el índice mientras se sigue usando el anterior

webir - Indexación Dinámica – Indice Auxiliar (IA)

- Indice auxiliar (IA) de documentos nuevos – cambios disponibles inmediatamente
- Incorporación rápida de nuevos documentos
- En memoria
- Búsquedas en los dos índices
- Vector de bits para documentos eliminados
  - Edición de docs = eliminación + reinserción
- Cuando el IA es muy grande – unificar a disco con el índice principal

webir - Indexación Dinámica – Indice Auxiliar (IA)

- Costo de la unificación depende de como se almacena el IA
  - Cada lista de postings en un archivo
    - Levantar el archivo y agregar los postings del IA
    - Para evitar múltiples accesos a disco, acumular cambios para actualizar un menor número de veces el índice principal (disco)
  - Todo el diccionario en un sólo archivo
    - Concatenación de las listas
      - Se recorre cada posting del índice en disco T/n veces, n es el tamaño del IA y T el número total de postings
      - $O(T^2/n)$

termID docID1 docID2 docID3 termID' docID1' docID2' docID3' termID" docID1" docID2" docID3" ...

    - Se generan varios IA de tamaños $2^0 \times n$, $2^1 \times n$, $2^2 \times n$, …, → **merge logarítmico**
      - Se recorre cada posting una vez en los log(T/n) niveles
      - O(T log(T/n))
      - Consultas más lentas y operaciones/estadísticas más complejas

termID docID3 termID' docID3' ...

termID docID1 docID2 termID' docID1' docID2' termID" docID1" docID2' ...'

termID docID11 docID12 termID' docID11' docID12' termID" docID1" docID12" ...

webir - Indexación Avanzada

- Construcción de índices posicionales
  - Se aplican los mismos algoritmos con pequeños cambios a ternas (termID, docID, (pos1, pos2,...))
- Recuperación ordenada por algún criterio (Ranking function)
  - Coeficiente, en forma descendente
  - Las recorridas se realizan solo de la primer parte de la listas
  - No es sólo inserción de docs nuevos al final, es necesario ordenar por el criterio

webir - Ejercicio

- ¿Puede la corrección ortográfica ser una fuente de inseguridad?

webir - Ejercicio

- ¿Puede la corrección ortográfica ser una fuente de inseguridad?
- ¿Si se toman en cuenta (términos de) docs a los que el usuario no tiene acceso?

webir - Seguridad

- Access Control Lists (ACL)
  - Indice invertido de los docs a los que puede acceder cada usuario
  - Estas listas se cruzan con las listas de resultados
  - Proceso lento para los usuarios con acceso a muchos docs
    - Muchas veces se obtienen los permisos del ”File System”

- “Misuse detection for information retrieval systems”, Cathey, Ma, Goharian, Grossman. Proceedings of the twelfth international conference on Information and knowledge management, 2003.
- “Using relevance feedback to detect misuse for information retrieval systems”, Ma, Goharian. Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004.
- “Query length impact on misuse detection in information retrieval systems”, Ma, Goharian. Proceedings of the 2005 ACM symposium on Applied computing, 2005.

webir - Compresión de Índices

- Menor uso de disco
- Mejor uso de cache
  - Términos frecuentes
- Más información en la memoria
  - Descomprimir a medida que sea necesario
- Menos accesos a disco

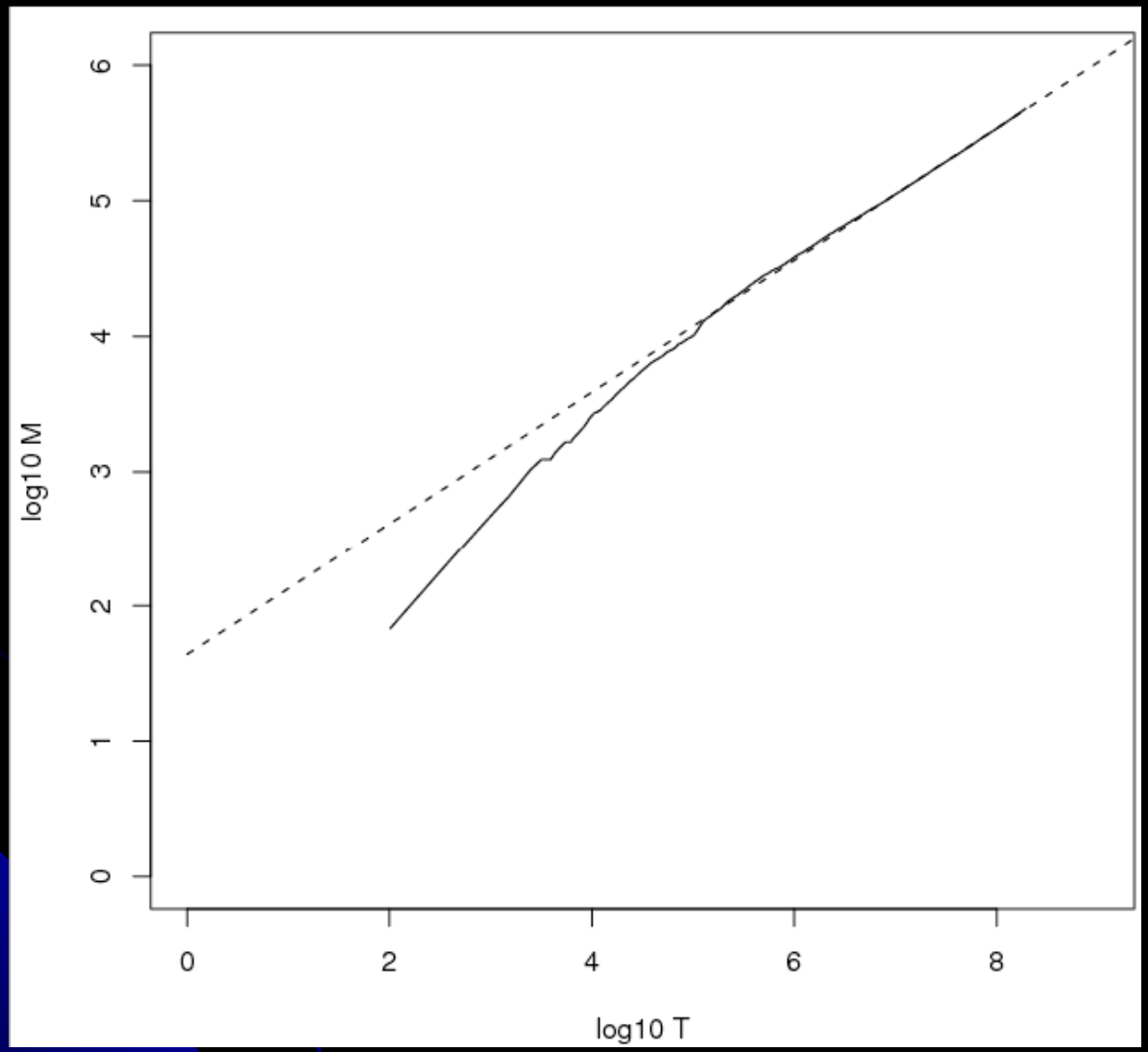
- Compresión del vocabulario
- Compresión de las listas de postings

- Ley de Heap
  - Estimación del tamaño del vocabulario (M) en función del tamaño de la colección (T) (tamaño de la colección = número de tokens)
  - M tiene una relación lineal con T en una representación log-log (*log M vs log T*)

$$M = k T^b$$

    - $b \sim 0.5$
    - $30 \leq k \leq 100$
  - Vocabulario sigue creciendo a medida que crece la colección
  - Vocabularios grandes para grandes colecciones
  - En el ejemplo de Reuters: b = 0.49 y k = 44

webir - Compresión de Indices – Ley de Heap



webir - Compresión de Índices

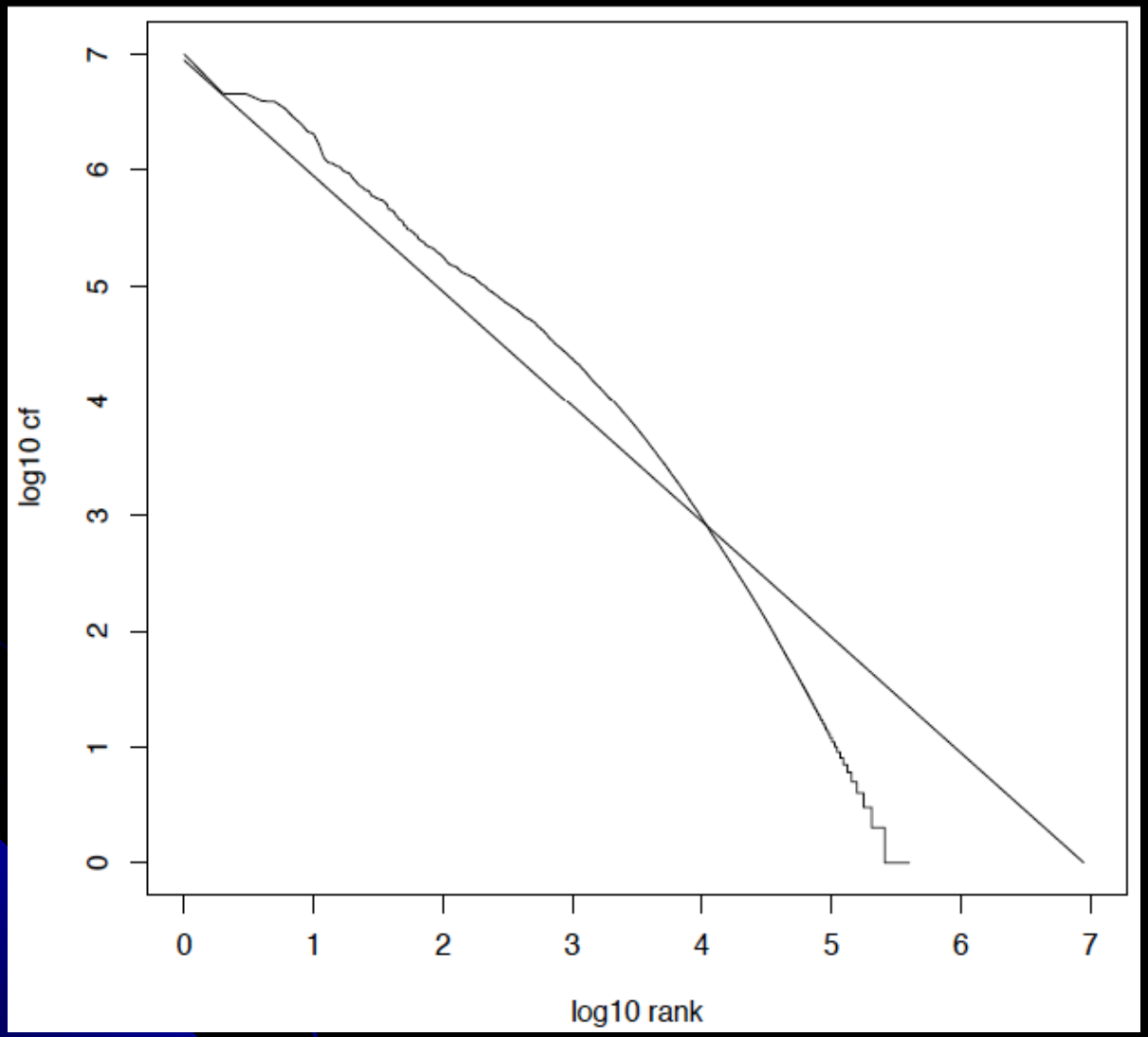
- Ley de Zipf
  - Un pequeño número de palabras son utilizadas con mucha frecuencia
  - Un gran número de palabras son poco empleadas
  - cf_i es la frecuencia de una palabra en la *i*-ésima posición
    - ordenadas de mayor a menor frecuencia

      cf_i *proporcional a* $1/i$
  - Relación lineal en una representación log-log (*log cf vs log rank*)

    $\log cf_i = \log c - \log i$

  - Importante para la compresión de las listas de postings

webir - Compresión de Indices – Ley de Zipf



webir - Compresión de Índices - Vocabulario

- El vocabulario es pequeño comparado con las listas de postings
- Aumentar la velocidad de recuperación
  - Menos accesos a disco
- Limitaciones de Hardware
  - Celulares, etc.
- Acelerar el tiempo inicial
- Dos técnicas para el diccionario en orden lexicográfico
  - En un solo string (por término) y por bloques

webir - Compresión de Índices - Vocabulario

- Ejemplo Reuters-RCV1
  - 1 GB de texto
  - Noticias de un año 1996-1997
  - Documentos N = 800 000
  - Tokens T = 100 000 000 (100 millones de palabras no normalizadas)
  - Tamaño del vocabulario M = 400 000 (391,523 de términos)
  - 4 bytes para cada termID = 0.8 GB
  - Largo prom. de las palabras = 8 bytes

webir - Compresión de Vocabulario

- Forma más simple de guardar términos: Array de celdas de igual tamaño (fijo)
- Por ejemplo 20 bytes por palabra
  - Orden alfabético
  - Búsquedas por bipartición

término	frequencia documento	Puntero a lista postings
a	656,265	→
aachen	65	→
...		
Zulu	221	→

Ejemplo Reuters → espacio necesario: 20 bytes 4 bytes 4 bytes

webir - Compresión de Vocabulario

- Ejemplo
  - 20 bytes para el término – promedio 8 bytes
  - 4 bytes para la frecuencia (número de docs en que aparece)
  - 4 bytes para el puntero a la lista de postings
  - Reuters-RCV1
    - M * (20+4+4) = 400 000 * 28 = 11.2 megabyte (MB)
    - Tamaño del vocabulario M = 400 000
    - Documentos N = 800 000
- Búsquedas por bipartición

webir - Compresión de Vocabulario

- Desventajas del array de celdas de igual tamaño (fijo)
- Desperdicio de espacio
- ¿Qué pasa con términos largos?

webir - Compresión de Vocabulario - Diccionario en un Solo String

- Lista continua de todos los términos
  - Reducción de hasta 60% vs espacio fijo para las palabras
  - Resuelve problemas de palabras más largas de lo esperado
- Se debe tener el puntero a donde comienza cada término
  - También indica donde termina el anterior
- Búsquedas por bipartición

webir - Compresión de Vocabulario - Diccionario en un Solo String

... arbolarteriaarterioesclerosisartrosis...

Frec	Postings	Comienzo
3	→	
49	→	
29	→	
17	→	

webir - Compresión de Vocabulario - Diccionario en un Solo String

- Ejemplo Reuters-RCV1
  - Para direccionar el espacio de los términos (largo prom. = 8 bits)
    - 400 000 * 8 = $3.2 * 10^6 \rightarrow \log_2 3.2 * 10^6$ = 22 bits = 3 bytes
  - 4 bytes para la frecuencia (número de docs en que aparece)
  - 4 bytes para el puntero a la lista de postings
  - M * (8+4+4+3) = 400 000 * 19 = 7.6 MB (11.2 MB)
  - 1/3

webir - Compresión de Vocabulario – Por Bloques

- Se puede comprimir aún más separando el vocabulario anterior en bloques de *k* términos

... 5Arbol7Arteria17Arterioesclerosis8Artrosis...

... 9Elemental8Elemento8Eliminar...

...

... 5Zorro6Zorzal8Zozobrar...

- Sólo se guarda el puntero al comienzo del bloque
- Se agrega un byte adicional a cada término para indicar su largo
- Se eliminan *k-1* punteros a términos por bloque

webir - Compresión de Vocabulario – Por Bloques

- Ejemplo Reuters-RCV1
  - k = 4 → se agregan 4 bytes para indicar largo, se eliminan (k-1)*3 = 9 bytes
  - Por cada bloque de 4 términos se ahorran 5 bytes → 400 000 * ¼ * 5 = 0.5 MB
  - Total 7.1 MB (7.6 MB)

webir - Compresión de Vocabulario – Por Bloques

- ¿Cómo se hacen las búsquedas?
- ¿Se puede mejorar la compresión aumentando el tamaño de los bloques?

webir - Compresión de Vocabulario – Por Bloques

- ¿Se puede mejorar la compresión aumentando el tamaño de los bloques?
- Aumenta el tiempo de búsqueda
- Búsqueda por bipartición de bloques y búsqueda secuencial dentro de cada bloque

webir - Compresión de Vocabulario – Por Bloques y Front Coding

- Orden alfabético permite que hayan muchas palabras consecutivas con prefijos en común
- Codificación Frontal – Front Coding

... 8Automata8Automate9automatic10Automation...

... 8Automat*a1¤e2¤ic3¤ion...

  - * marca el final del prefijo común
  - ¤ reemplaza prefijo en los términos siguientes
  - Se indica el largo de cada sufijo o prefijo
- Ejemplo Reuters-RCV1
  - Se ahorran 1.2 MB
  - Total 5.9 MB